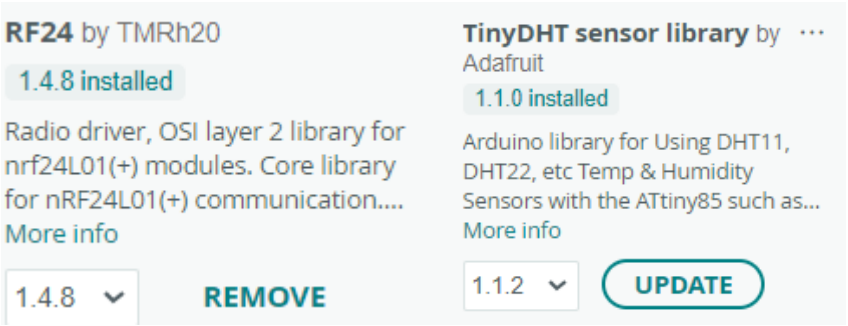
Rapport technique du SAE
Station météo maison

Librairies Arduino:



Programme Arduino extérieur :

```
#include <SPI.h>
#include <RF24.h>
#include "TinyDHT.h"

#define DHTPIN 6 // SDA, or almost any other I/O pin
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
float t;
float h;
float message;

#define pinCE   7              // On associe la broche "CE" du NRF24L01 à la sortie digitale D7 de l'arduino
#define pinCSN   8             // On associe la broche "CSN" du NRF24L01 à la sortie digitale D8 de l'arduino
#define tunne2  "PIPE0"        // On définit un "nom de tunnel" (5 caractères), pour pouvoir communiquer d'un NRF24 à l'autre

RF24 radio(pinCE, pinCSN);     // Instanciation du NRF24L01

const byte adresse[6] = tunne2;                // Mise au format "byte array" du nom du tunnel

void setup() {
  dht.begin();
  radio.begin();                       // Initialisation du module NRF24
  Serial.begin(9600);
  radio.openWritingPipe(adresse);      // Ouverture du tunnel en ÉCRITURE, avec le "nom" qu'on lui a donné
  radio.setChannel(5);
  radio.setPALevel(RF24_PA_MIN);       // Sélection d'un niveau "MINIMAL" pour communiquer (pas besoin d'une forte puissance, pour nos essais)
```

```
  radio.stopListening();              // Arrêt de l'écoute du NRF24 (signifiant qu'on
va émettre, et non recevoir, ici)
}

void loop() {
  h=dht.readHumidity();
  t=dht.readTemperature();
  radio.write(&t, sizeof(t));     // Envoi de notre message
  radio.write(&h, sizeof(h));     // Envoi de notre message
  delay(1000);
  Serial.print("h=");Serial.print(h,1);Serial.print("\t");
  Serial.print("t=");Serial.println(t,1);
  delay(2000); //Collecting period should be : >1.7 second
}
```

Programme Arduino intérieur :

```
#include <NTPClient.h>
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>
#include <SPI.h>
#include <RF24.h>
#include <TimeLib.h> // Ajout de la bibliothèque pour manipuler le temps

#define DHTPIN 33
#define DHTTYPE DHT22

#define WIFI_SSID ""
#define WIFI_PASSWORD ""
#define API_KEY "AIzaSyD_K8tPWnsfffhYnoqfhsQg54Z5jQxh1FY"
#define FIREBASE_PROJECT_ID "esp32dht22-5ae67"
#define USER_EMAIL "enzoborowec@gmail.com"
#define USER_PASSWORD "Vgaming0301"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


#define pinCE   4
#define pinCSN  5
#define tunne2  "PIPE0"

DHT dht(DHTPIN, DHTTYPE);
```

```
WiFiUDP ntpUDP; // Instance de l'objet WiFiUDP pour NTPClient
NTPClient timeClient(ntpUDP, "pool.ntp.org", 3600);

RF24 radio(pinCE, pinCSN);

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

unsigned long dataMillis = 0;

void setup() {
    Serial.begin(9600);
    Serial.println("Récepteur NRF24L01");
    Serial.println("");
    dht.begin();
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    config.api_key = API_KEY;
    auth.user.email = USER_EMAIL;
    auth.user.password = USER_PASSWORD;

    Firebase.begin(&config, &auth);
    Firebase.reconnectWiFi(true);

    timeClient.begin();

    radio.begin();
    radio.openReadingPipe(1, (const byte *)tunne2);
    radio.setPALevel(RF24_PA_MIN);
    radio.setChannel(5);
    radio.startListening();
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  // Inverser l'affichage
```

```
  display.setRotation(2);

  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  // Dessiner les barres du signal WiFi dans les 6 premiers pixels jaunes
  for (int i = 0; i < 6; i++) {
    int barHeight = map(i, 0, 5, 1, 6); // Taille de la barre en fonction de sa
position
    for (int j = 0; j < barHeight; j++) {
      display.drawPixel(i * 2, 5 - j, SSD1306_WHITE); // Dessiner les barres avec un
espace de 1 pixel entre elles
    }
  }

  // Ajouter un texte sous la barre de réseau
  String message = "Hello World";
  int textWidth = message.length() * 6; // Chaque caractère est de 6 pixels de large
  int cursorX = (SCREEN_WIDTH - textWidth) / 2; // Centrer le texte
  display.setCursor(cursorX, 20); // Déplacer le curseur à la position calculée
  display.println(message); // Afficher le texte

  display.display();
}

void loop() {
    timeClient.update();

    // Vérifiez si Firebase est prêt et s'il est temps de collecter de nouvelles
données
    if (Firebase.ready() && (millis() - dataMillis > 60000 || dataMillis == 0)) {
        dataMillis = millis();

        // Lecture des données de température et d'humidité intérieures
        float humidity = dht.readHumidity();
        float temperature = dht.readTemperature();

        Serial.println(temperature);
        Serial.println(humidity);

        if (!isnan(humidity) && !isnan(temperature)) {
            // Création du document pour l'intérieur
            FirebaseJson content;
            String timestamp = timeClient.getFormattedTime(); // Obtient la date et
l'heure au format RFC 3339
            timestamp.replace("Z", ""); // Supprime le "Z" à la fin
```

```
            // Ajoute le décalage horaire (3600 secondes = 1 heure)
            time_t utc = timeClient.getEpochTime() + 3600;
            struct tm timeinfo;
            gmtime_r(&utc, &timeinfo);
            char strftime_buf[64];
            strftime(strftime_buf, sizeof(strftime_buf), "%FT%TZ", &timeinfo);
            timestamp = String(strftime_buf);

            content.set("fields/Temperature/stringValue", String(temperature));
            content.set("fields/Humidity/stringValue", String(humidity));
            content.set("fields/Timestamp/stringValue", timestamp); // Utilisez
stringValue pour le format RFC 3339

            String documentPathInside = "intérieur/" + timestamp;

            // Création du document pour l'intérieur dans Firestore
            if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, "",
documentPathInside.c_str(), content.raw())) {
                Serial.println("Document intérieur créé avec succès");
            } else {
                Serial.println("Échec de création du document intérieur");
                Serial.println(fbdo.errorReason());
            }
        } else {
            Serial.println("Échec de lecture des données DHT");
        }

        // Lecture des données de température et d'humidité extérieures via NRF24L01
        if (radio.available()) {
            float temperatureExt;
            float humidityExt;
            radio.read(&temperatureExt, sizeof(temperatureExt)); // Lecture de la
température extérieure
            radio.read(&humidityExt, sizeof(humidityExt)); // Lecture de l'humidité
extérieure
            Serial.print("Température extérieure reçue : ");
            Serial.println(temperatureExt);
            Serial.print("Humidité extérieure reçue : ");
            Serial.println(humidityExt);

            // Création du document pour l'extérieur
            FirebaseJson contentExt;
            String timestampExt = timeClient.getFormattedTime(); // Obtient la date
et l'heure au format RFC 3339
            timestampExt.replace("Z", ""); // Supprime le "Z" à la fin

            // Ajoute le décalage horaire (3600 secondes = 1 heure)
```

```
            time_t utcExt = timeClient.getEpochTime() + 3600;
            struct tm timeinfoExt;
            gmtime_r(&utcExt, &timeinfoExt);
            char strftime_bufExt[64];
            strftime(strftime_bufExt, sizeof(strftime_bufExt), "%FT%TZ",
&timeinfoExt);
            timestampExt = String(strftime_bufExt);

            contentExt.set("fields/Temperature/stringValue", String(temperatureExt));
            contentExt.set("fields/Humidity/stringValue", String(humidityExt));
            contentExt.set("fields/Timestamp/stringValue", timestampExt); // Utilisez
stringValue pour le format RFC 3339

            String documentPathOutside = "extérieur/" + timestampExt;

            // Création du document pour l'extérieur dans Firestore
            if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, "",
documentPathOutside.c_str(), contentExt.raw())) {
                Serial.println("Document extérieur créé avec succès");
            } else {
                Serial.println("Échec de création du document extérieur");
                Serial.println(fbdo.errorReason());
            }
            display.clearDisplay();
        display.setCursor(0, 0);
        display.print("Temperature: ");
        display.print(temperature);
        display.println(" C");
        display.print("Humidity: ");
        display.print(humidity);
        display.println(" %");
        display.print("TemperatureExt: ");
        display.print(temperatureExt);
        display.println(" C");
        display.print("HumidityExt: ");
        display.print(humidityExt);
        display.println(" %");
        display.display();
        }
    }

    delay(60000);
}
```

Programme VSCode :

```
<!DOCTYPE html>
<html lang="fr">
```

```
<head>
  <meta charset="utf-8"/>
  <title>Température et humidité</title>
  <style>
    * {
      text-align: center !important;
      margin-left: auto !important;
      margin-right: auto !important;
    }
    .titre {
        border-top: 0px;
        text-align: center;
        font-size: xx-large;
    }
    #toggle {
        display: none;
      }

      label {
        display: inline-block;;
        width: 40px;
        height: 20px;
        background-color: #ccc;
        border-radius: 10px;
        position: relative;
        cursor: pointer;
        margin-left: 32px;
      }

      label::before {
        content: "";
        position: absolute;
        top: 2px;
        left: 2px;
        width: 16px;
        height: 16px;
        background-color: white;
        border-radius: 50%;
        transition: transform 0.3s;
      }

      #toggle:checked + label {
        background-color: #2196F3;
      }

      #toggle:checked + label::before {
        transform: translateX(20px);
      }
    table, th, td{
        border: 2px solid black;
        border-collapse: collapse;
    }
  </style>
</head>

<body>
  <div id="titre" class="titre"><p>Capteurs maison</p></div>
```

```
  <p id="texte2">Contrôle lumière</p>
  <input type="checkbox" id="toggle" class="checkbox"/>
  <label for="toggle" class="toggle"></label>
  <p id="texte">Dernière mesure :</p>
  <input id='myInput' onkeyup='searchTable()' type='text' placeholder="Rechercher une valeur"
style="margin-bottom: 30px">
  <table class="table table-bordered" id="myTable">
    <thead>
        <tr>
            <th scope="col">Date et heure <br/>(AAAA-MM-JJ HH:MM:SS)</th>
            <th scope="col">Température intérieure</th>
            <th scope="col">Humidité intérieure</th>
            <th scope="col">Température extérieure</th>
            <th scope="col">Humidité extérieure</th>
        </tr>
    </thead>
    <tbody>
        <tr>
        </tr>
    </tbody>
</table>

  <script type="module">
    import { initializeApp } from "https://www.gstatic.com/firebasejs/10.9.0/firebase-app.js";
    import { getAnalytics } from "https://www.gstatic.com/firebasejs/10.9.0/firebase-analytics.js";
    import { getFirestore } from "https://www.gstatic.com/firebasejs/10.9.0/firebase-firestore.js";

    const firebaseConfig = {
      apiKey: "AIzaSyD_K8tPWnsfffhYnoqfhsQg54Z5jQxh1FY",
      authDomain: "esp32dht22-5ae67.firebaseapp.com",
      databaseURL: "https://esp32dht22-5ae67-default-rtdb.europe-west1.firebasedatabase.app",
      projectId: "esp32dht22-5ae67",
      storageBucket: "esp32dht22-5ae67.appspot.com",
      messagingSenderId: "740102306",
      appId: "1:740102306:web:4037c23103e7a935c5c7fb",
      measurementId: "G-E5RCY0ZQJT"
    };
    const app = initializeApp(firebaseConfig);
    const analytics = getAnalytics(app);
    const db = getFirestore(app);

    import { query, where, doc, getDocs, collection, addDoc, onSnapshot, setDoc } from
"https://www.gstatic.com/firebasejs/10.9.0/firebase-firestore.js";

    function lecture(collec,docu,champ,texte,id) {
      const unsub1 = onSnapshot(doc(db,collec,docu), (doc)=>{
        const data = doc.data();
        let formattedData = "";
        for (const key in data) {
          if (key==champ) {
            formattedData += texte;
            formattedData += data[key];
          }
        }
        document.getElementById(id).innerHTML = formattedData;
      })
    }
```

```
const q = query(collection(db, "intérieur"));
const querySnapshot = await getDocs(q);
let donnees = "";
querySnapshot.forEach((doc) => {
  var table = document.getElementById("myTable");
  var row = table.insertRow(2);
  let cell1 = row.insertCell(0);
  var cell2 = row.insertCell(1);
  var cell3 = row.insertCell(2);
      var cell4 = row.insertCell(3);
      var cell5 = row.insertCell(4);
  donnees = doc.data();
  for (const key in donnees) {
    if (key=="Timestamp") {
    cell1.innerHTML = donnees[key];
    }
    if (key=="Temperature") {
    cell2.innerHTML = donnees[key];
    }
    if (key=="Humidity") {
    cell3.innerHTML = donnees[key];
    }
  }
});
const q2 = query(collection(db, "extérieur"));
const querySnapshot2 = await getDocs(q2);
let i = 2, j = 0;
let donnees2 = "";
querySnapshot2.forEach((doc) => {
  j = 0;
  donnees2 = doc.data();
  var table = document.getElementById("myTable");
  for (i=2;i<table.rows.length;i++) {
    if (table.rows[i].cells[0].innerHTML==donnees2["Timestamp"]) j = i;
  }
  for (const key in donnees2) {
    if (j != 0) {
      var row = table.rows[j];
      if (key=="Temperature") {
        table.rows[j].cells[3].innerHTML = donnees2[key];
      }
      if (key=="Humidity") {
        table.rows[j].cells[4].innerHTML = donnees2[key];
      }
    }
  }
  i++;
});

  //cell1.innerHTML = lecture("intérieur","2024-04-03T11:55:55Z","Temperature","");

//lecture("intérieur","2024-04-03T12:03:02Z","Timestamp","Moment mesure : ","val1");
//lecture("intérieur","2024-04-03T12:03:02Z","Temperature","Température intérieure : ","val2");
//lecture("intérieur","2024-04-03T12:03:02Z","Humidity","Humidité intérieure : ","val3");
//lecture("extérieur","2024-04-03T12:03:02Z","Temperature","Température extérieure : ","val4");
//lecture("extérieur","2024-04-03T12:03:02Z","Humidity","Humidité extérieure : ","val5");
```

```
    const slider = document.getElementById("toggle");
    slider.addEventListener("change",function() {
      if (slider.checked) {
        const docRef = setDoc(doc(db,"Lampe","Lampe"), {
          state : true
        })
      }
      else {
        const docRef = setDoc(doc(db, "Lampe", "Lampe"), {
          state : false
        })
      }
    });
  </script>

  <script type="">
    function searchTable() {
        var input, filter, found, table, tr, td, i, j;
        input = document.getElementById("myInput");
        filter = input.value.toUpperCase();
        table = document.getElementById("myTable");
        tr = table.getElementsByTagName("tr");
        for (i = 1; i < tr.length; i++) {
            td = tr[i].getElementsByTagName("td");
            for (j = 0; j < td.length; j++) {
                if (td[j].innerHTML.toUpperCase().indexOf(filter) > -1) {
                    found = true;
                }
            }
            if (found) {
                tr[i].style.display = "";
                found = false;
            } else {
                tr[i].style.display = "none";
            }
        }
    }
  </script>
</body>
</html>
```